Real and/or Complex Roots Calculation of Nonlinear Equations Systems through Modified Particle Swarm Optimization[1]

Cálculo de raíces reales y/o complejas de sistemas de ecuaciones no lineales mediante el método de enjambre de partículas modificado[2]

Cálculo de raízes reais e/ou complexas em sistemas de equações não lineares pelo método de enxame de partículas modificado[3]

Sergio Reyes-Sierra[4]
Julián Plata-Rueda[5]
Rodrigo Correa-Cely[6]

SICI: 0123-2126(201212)16:2<349:RCRCNE>2.0.TX;2-R

[1] Submitted on: March 11, 2011. Accepted on: February 15, 2012. This article is derived from a research modality graduation paper, as part of the *Simulation Tools Development Project*. Internal financing. Developed by the research group *Control, Electronics, Modelling and Simulation, CEMOS of Universidad Industrial de Santander,* Bucaramanga, Colombia.

[2] Fecha de recepción: 12 de marzo de 2011. Fecha de aceptación: 15 de febrero de 2012. Este artículo se deriva del proyecto de Trabajo de grado modalidad investigación como parte del proyecto de desarrollo de herramientas de simulación. Financiación interna. Desarrollado por el grupo de investigación Control, Electrónica, Modelado y Simulación (CEMOS), de la Universidad Industrial de Santander, Bucaramanga, Colombia.

[3] Data de recebimento: 12 de março de 2011. Data de aceite: 15 de fevereiro de 2012. Este artigo é derivado do projeto de Trabalho de graduação modalidade pesquisa como parte do projeto de desenvolvimento de ferramentas de simulação. Financiamento interno. Desenvolvido pelo grupo de pesquisa Controle, Eletrônica, Modelado e Simulação (CEMOS), da Universidade Industrial de Santander, Bucaramanga, Colômbia.

[4] Ingeniero electrónico, Universidad Industrial de Santander, Bucaramanga, Colombia. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones, Universidad Industrial de Santander. Bucaramanga, Colombia. Correo electrónico: sergio2052144@hotmail.com.

[5] Ingeniero electrónico, Universidad Industrial de Santander, Bucaramanga, Colombia. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones, Universidad Industrial de Santander. Bucaramanga, Colombia. Correo electrónico: julianplatar@live.com.

[6] Ingeniero químico, Universidad Nacional de Colombia, Bogotá, Colombia. Magíster en Ingeniería Química, Universidad Industrial de Santander, Bucaramanga, Colombia. Magíster en Ingeniería Química, Lehigh University, Estados Unidos. Doctorado en Polymer Science and Engineering, Lehigh University. Profesor titular, Universidad Industrial de Santander. Bucaramanga, Colombia. Correo electrónico: crcorrea@uis.edu.co.

Resumen

En este artículo se describe una alternativa numérica para solucionar sistemas de ecuaciones no lineales con raíces reales y/o complejas. Para ello se convirtió el problema de solución directa de tales sistemas en un problema de optimización, y se resolvió utilizando el método de enjambre de partículas apropiadamente modificado para tal tarea. A título demostrativo se incluyen algunos resultados con sistemas de dos, cinco y diez ecuaciones, resueltos en un computador convencional y en un arreglo de cuatro nodos. Se concluyó que la estrategia es válida para solucionar este tipo de sistemas. Por otra parte, no se detectó una mejora en los tiempos de computación cuando se utilizó el *cluster*.

Palabras clave

Optimización mediante enjambre de partículas, sistemas de ecuaciones no lineales, metaheurística, *cluster*.

Abstract

In this article we describe a numerical alternative to sovlve systems of nonlinear equations with real and/or complex roots. The problem of solving directly such systems was transformed into an optimization one, which was solved using a specially modified particle swarm method. As an example, system of two, five and ten equations were solved using a conventional personal computer as well as a cluster of four nodes. It was concluded that this strategy is valid for solving this type of systems of equations. Moreover, using the cluster no computational time improvement was detected.

Key words

Particle swarm optimization, nonlinear system of equations, metaheuristic, cluster

Resumo

Este artigo descreve uma alternativa numérica para resolução de sistemas de equações não lineares com raízes reais e/ou complexas. Para isso, o problema de solução direta de tais sistemas virou-se para um problema de otimização e resolveu-se utilizando o método de enxame de partículas devidamente modificado para esta tarefa. A maneira de demonstração incluem-se alguns resultados com os sistemas de dois, cinco e dez equações, resolvidos em computador convencional e num arranjo de quatro nós. Concluiu-se que a estratégia é válida para resolver este tipo de sistema. Além disso, nenhuma melhoria foi detectada nos tempos de computação quando o *cluster* foi utilizado.

Palavras-chave

Utilização mediante enxame de partículas, sistemas de equações não lineares, meta-heurística, *cluster*.

Introduction

Given the difficulty for representing physical phenomena through one or more non-linear systems of equations (NSE), it is necessary to have several analytic and/or numerical methods available for solving them (Grosan and Abraham, 2008; Bianchini et al., 2001; Floudas, 1999; Ortega and Rheinboldt, 1970). Some of these systems do not have a single root (solution), so finding them becomes a mathematical and computational challenge, which is, as of today, an open research topic. In some cases, algebraic approaches can be used to find some of them. However, most NSE are too difficult to solve in this way, so numerical approaches are used to obtain approximate solutions, with a given error margin (Luo *et al*., 2008). One of the most used methods is multidimensional Newton Raphson (NR). In spite of its simple algorithm and high convergence speed, its main weakness is that it is dependent on a user-defined starting point, which needs to be close to the desired solution, meaning that it requires a previous knowledge of the solution. When a system with several variables and relatively complex equations is to be solved, the near-solution starting point becomes almost impossible to guess. Another traditional approach is the method of the gradient. This, however, requires that the system complies with a differentiability condition, which can prove to be difficult to achieve. Through past years, the so-called evolutionary algorithms have become a powerful choice for numerical solution (Geng *et al*., 2009; Hatanaka *et al*., 2004; Yang et al.,2008). When the NSE gets bigger, traditional approaches become excessively difficult and non-practical, increasing the computational cost, thus being necessary to use more recent methods (Brits *et al*., 2002; Hui and Zhao, 2008; Grosan and Abraham, 2008; Cui and Cai, 2010).

During the past decades, the metaheuristic optimization approaches (based on the imitation of natural, biologic, social or cultural processes) have been successfully applied to several optimization problems on engineering. One of these techniques is Particle Swarm Optimization (PSO), which has a high ability to

efficiently explore multidimensional search spaces (Aijia *et al*., 2009; Rao, 2009; Clerc, 2006), and which has been recently applied to the current problem, e.g. (Tsoulos *et al*., 2010; Ouyang *et al*., 2009).

This article describes the development of an algorithm that allows solving NSE with real and/or complex roots, based on PSO. Striving to lower the computation time and the required memory, a theorem is presented that allows the transformation of a finding roots problem into an optimization one, and its demonstration is presented later on. Moreover, some results for a couple of demonstrative examples, using a traditional computer and a four node cluster, are shown.

1. Fundamentals

An NSE with m functions and m unknowns is assumed, as shown by equation (1), where $F_i(z_1,z_2,\ldots,z_i)$, $i = 1,2,\ldots,m$ are the non-linear equations, while z_i, $i = 1,2,\ldots,m$, are the complex variables.

$$\begin{aligned} &F_1(z_1,z_2,\ldots,z_m)\\ &F_2(z_1,z_2,\ldots,z_m)\\ &\vdots\\ &F_m(z_1,z_2,\ldots,z_m) \end{aligned} \tag{1}$$

Even though there are several numerical strategies to solve this type of systems, it was decided to implement the following theorem, to transform it into an optimization problem (minimization) (Ortega and Rheinboldt, 1970; Gómez, 2010). This theorem is also valid for the case where all the solutions are in the domain of the real numbers.

Theorem 1: Let $\mathbb{C}$ be the set of complex numbers, X a subset of $\mathbb{C}^n$ and consider the following system of equations:

$$System\begin{Bmatrix} f1(z)=0\\ f2(z)=0\\ \quad\vdots\quad with\ z=(z_1,z_2,...,zn)\\ fm(z)=0 \end{Bmatrix} \tag{2}$$

Where, for each i, f_i is a function whose domain contains X, and its range belongs to the complex numbers. Let $f : X \to \mathbb{R}$ be defined as:

$$f(z)=\sum_{i=1}^{m} fi(z)\overline{f_i(z)}=\sum_{i=1}^{m}\| fi(z)\|^2 \tag{3}$$

With $z = (z_1, z_2, \ldots, z_n)$ and where $\| c \|$ represents the magnitude of the complex number c, $\bar{c}$ is its complex conjugate and $\| c \|^2 = c\bar{c}$. Note that f is properly defined, and, besides, the images of the function are non-negative real numbers. Thus, it follows that:

Proposition 1, Suppose that the system (2) has a solution in X, and let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in X$. Therefore:

α satisfies (2) if, and only if, α minimizes f.

Proof. If α satisfies (2) then $f_i(\alpha) = 0$ for every $i = 1,2,\ldots,m$. Therefore $f(\alpha) = 0$ and, since $f(z) \geq 0$ for all $z \in X$, then α is a minimum for f.

Now, if α minimizes f but does not satisfies (2) then $f(\alpha)$ must be positive, since $f(z) \geq 0$ for all $z \in X$. Since the system has a solution in X, there is a $z^* \in X$ such that $f(z^*) = 0$ y $z^* \neq \alpha$. Therefore, $f(z^*) < f(\alpha)$ which violates α being a minimum for f. Note the importance on the general consistency condition over the system (2) in X, since given a system of equations, it is always possible to build f and, if α minimizes it, it does not generally imply that the system has a solution. Therefore, the problem of finding an NSE roots in a given set X, can be transformed into an optimization problem (minimization for this case), for a function f (built as previously shown) in the set X. An algorithm based on the previously stated is:

Algorithm 1.
Input: The NSE (2) and the set X
Step 1: Build f
Step 2: Minimize f over X.
Step 3: Let $\alpha \in X$ be a minimum point for f. If $f(\alpha) = 0$ then α satisfies (2). Otherwise (2) does not have a solution on X.

It is important to remark that it is not mandatory to use the squared magnitudes of each expression on the NSE; the magnitude itself can be used without any problem. A brief fundamentals of metaheuristic optimization algorithms is presented below.

Particle Swarm Optimization (PSO): this stochastic, adaptive optimization technique was developed by Eberhart and Kennedy in 1995. It is directly related to the ability that have individual members of a group to gain and share knowledge of their surroundings. From the analysis of this behavior, a simple mathematical model was obtained, which reveal its potential to solve optimization problems (Rao, 2009). This algorithm generates a population with

a given initial speed and position, which relates to a possible solution. For an N-dimensional problem, the position and speed are given by M x N matrices, as shown by equations (4) and (5).

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} \tag{4}$$

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1N} \\ v_{21} & v_{22} & \cdots & v_{2N} \\ \vdots & \vdots & & \vdots \\ v_{M1} & v_{M2} & \cdots & v_{MN} \end{bmatrix} \tag{5}$$

Where X and V are the position and speed matrices, respectively. The population is given by a set of M particles. Each row in the X matrix represents the position of a particle in the search space (Rao, 2009; Clerc, 2006). Each iteration, the particles memorize and follow their best position (Pbest), and the whole population best position (Gbest), to update the speed matrix. Pbest is defined as shown by equation (6).

$$Pbest = \begin{bmatrix} pbest_{11} & pbest_{12} & \cdots & pbest_{1N} \\ pbest_{21} & pbest_{22} & \cdots & pbest_{112N} \\ \vdots & \vdots & & \vdots \\ pbest_{M1} & pbest_{M2} & \cdots & pbest_{MN} \end{bmatrix} \tag{6}$$

Gbest, on the other hand, is the best position of the whole swarm, and is defined by equation (7).

$$Gbest = [gbest_1 \quad gbest_2 \quad \ldots \quad gbest_N] \tag{7}$$

With these results, the position and speed of the particles can be updated, using equations (8) y (9).

$$\begin{aligned} V_{ij}^{iter+1} &= w * V_{ij}^{iter} \\ &+ C1 * Rand(\,) * (pbest_{ij} - X_{ij}^{iter}) \\ &+ C2 * rand(\,) * (gbest_i - X_{ij}^{iter}) \end{aligned} \tag{8}$$

$$X_{ij}^{iter+1} = X_{ij}^{iter} + V_{ij}^{iter+1} \tag{9}$$

Where $i = 1,2,\ldots,M$ and $j = 1,2,\ldots,N$; *iter* is the current iteration; w is an inertia factor that regulates the effect of previous speeds into the new one;

*C*1 is a self-trust factor, while C2 relates to the social trust. *Rand( )* and *rand ( )* are functions that return an uniformly distributed random number between [0,1], and they weigh in the regulations of the individual and social information for each particle.

In order to apply the algorithm, some steps need to be followed (Rao, 2009, Parsopoulos and Vrhatis, 2010):

1. Provide a random value for the initial position and speed of the particles.
2. Evaluate the objective function, thus obtaining Pbest y Gbest.
3. Update the speed and position of each particle, according to equations (8) and (9).
4. Evaluate the objective function.
5. Compare, for each particle, the actual value of the function and that of Pbest; if said value is better, then update Pbest.
6. Select, from the current iteration, the particle with the best value of the objective function and compare them with Gbest. In case the former has a better value, then update Gbest.
7. Compare the value of the function on Gbest; if it does not comply with the stop criteria, then return to 3.

Algorithm modification: In order to calculate complex roots, it was necessary to modify the original algorithm, specifically in the update of speeds (8). The new equation is given by (10).

$$\begin{aligned} V_{ij}^{iter+1} &= w * V_{ij}^{iter} \\ &+ C1 * rand(\,) * real\left(pbest_{ij} - X_{ij}^{iter}\right) \\ &+ C2 * Rand(\,) * real\left(gbest_{i} - X_{ij}^{iter}\right) \\ &+ C1 * rand(\,) * imag\left(pbest_{ij} - X_{ij}^{iter}\right) \\ &+ C2 * Rand(\,) * imag\left(gbest_{i} - X_{ij}^{iter}\right) \end{aligned} \tag{10}$$

Where *real*(…) is the real part and *imag*(…) the imaginary one. These particles allow for movement in the whole complex plane.

2. Experiments

Experiments were carried out in both, a regular computer and a four node cluster, with the following specifications:

1. Regular computer (C1): Dell XPS 16.
   a. Processor: Intel Core i5 M430 @2,27 GHz, turbo mode 2,53 GHz; 3 MB cache.
   b. Ram: 4,0 GB DDR3.
   c. OS: Windows 7, 64 bits.
2. *Cluster* (C2):
   a. Nodes: 4, each one with 2 GB RAM.
   b. Processor: 3,2 GHz Intel Pentium IV each.
   c. OS: Rocks.
   d. Network Interface: Gigabit Ethernet.
   e. Cores by node: 2.

Three repetitions were carried out for each experiment, and a precision of $1 * 10^{-8}$ for the solution was used.

3. Results and Analysis

Following are some results, related to the solution of a NSE of two, five and ten equation, striving to show the method validity. Some other tests were also performed, but due to space limitations are not shown. Given these systems complexity, no analytical tool foretells how many real and/or complex roots they have, in case they even exist. Table 1 shows the real roots for the NSE (11).

$$F(x_a,x_b)=x_a*sen(x_b)+x_b*\cos(x_a)+10$$
$$G(x_a,x_b)=x_b*\cos(sen(x_b))+x_a*x_b*\cos(x_a)-2 \quad (11)$$

There, a set of solutions that satisfy the system is shown. The first couple is real and the rest is imaginary. For this case, 1000 particles were used, and the solution was found in a relative short time.

Table 1. Two equations, 1000 particles

Iter.	Time	Result	Function evaluation
409	4 s	Xa= -42.23465262154	F= -5.18435605556e-009
		Xb= 0.23802206697	G=-8.20914891619e-009
413	5 s	Xa= -2.1720584896 - 0.3202166466i	F=-5.7514686347e-009+6.4543450584e-009
		Xb= 1.5166862958 - 2.0403819021i	G= -7.5909130093e-009+3.9577201516e-009
450	5 s	Xa=-2.1976619256 + 0.1400807576i	F= 6.40819663999e-009-5.15596143557e-009
		Xb= 1.7985527695 - 2.1199359146i	G=-9.5202339345e-009+2.7501507738e-009

Source: Authors' own presentation.

3.1. Five equations system

The NSE (12) was implemented in computer C1; some results are shown in tables 2 and 3. Only the complex roots are reported.

$$
\begin{aligned}
F(x_a, x_b, x_c, x_d, x_e) &= x_a * \operatorname{sen}(x_b) + x_c^{x_d} - 32 * x_e \\
&= 3 * x_a + 8 * x_b + x_c \\
&+ x_c * \cos(x_b) - x_d + x_e - 143 \\
&+ 100 + 3 * x_d + 10 * x_e - 9 + x_d^{\cos(x_e)} \\
&+1 - 67 * \cos(x_b) + 5 * x_a^{x_d} \\
&- x_d * \operatorname{sen}(x_d) + 1
\end{aligned}
\qquad
\begin{aligned}
G(x_a, x_b, x_c, x_d, x_e) &= x_c * e^{x_a} + 3 * x_b + 6 * x_a \\
H(x_a, x_b, x_c, x_d, x_e) &= x_a + 8 * x_b - 3 * x_c + x_d \\
I(x_a, x_b, x_c, x_d, x_e) &= 3x_a - 6 * x_b + 8 * x_c
\end{aligned}
\qquad (12)
$$

Table 2. Five equations, 10 particles

Iter.	Time	Result	Function evaluation
2724	1 s	Xa= 0.9010426976 – 2.5549804799i	F= -2.4421740186e-008 -6.8860336988e-008i
		Xb= 8.0196175228 – 1.1627655589i	G= -6.2974656601e-008 -7.1980167604e-008i
		Xc= 27.3026436355 -37.3264351887i	H= -6.2010384827e-008 +5.1537514700e-008i
		Xd= -3.5454658212 – 4.7593944907i	I= 3.5227852855e-009 -4.3619332329e-008i
		Xe= 0.0677568460 – 0.1315619409i	J= 8.3611809032e-008 -1.1836561953e-008i
2526	1 s	Xa= 0.4258045313 – 5.0253757667i	F= -3.8404962932e-008 -4.4152377754e-008i
		Xb= 7.9215587328 + 1.1316100959i	G= 3.4654753733e-008 +9.3554692882e-008i
		Xc= 17.7095929914 +31.2190069696i	H=-3.0148044061e-009 +6.0320312079e-008i
		Xd= -19.4130567504 – 3.2254582879i	I= 7.8139658367e-008 +3.1166961403e-008i
		Xe= 0.0079927059 – 0.2694261176i	J= -7.2678119523e-008 +1.3231897356e-008i

Source: Authors' own presentation.

Table 3. Five equations, 300 particles

Iter.	Time	Result	Function evaluation
825	6 s	Xa= 2.2449516463 - 3.7341905832i	F= -3.8801642343e-008 -6.83399126178e-008i
		Xb= 13.0650469980 + 0.4456976578i	G= -1.6224888100e-008 -9.6703322239e-008i
		Xc= 11.8494402503 + 4.1722507485i	H= 8.5727780430e-009 -1.3196537196e-008i
		Xd= -7.4348952038 - 2.0739670195i	I= -7.4121829817e-008 -2.4495753639e-008i
		Xe= 0.0841438792 - 0.0330677414i	J= -2.5254713876e-009 -3.3119061271e-008i
819	6 s	Xa= 2.2449516448 + 3.7341905926i	F= 5.4868027011e-008 -7.4135003914e-008i
		Xb= 13.0650470039 - 0.4456976635i	G= 2.9225383003e-008 -7.2263476314e-008i
		Xc= 11.8494402425 - 4.1722508546i	H= -1.4341372534e-009 -3.0395197470e-010i
		Xd= -7.4348951871 + 2.0739670331i	I= 6.6720902247e-008 -4.05145164987e-008i
		Xe= 0.0841438774 + 0.0330677466i	J= -8.3087812186e-009 -7.5232996721e-008i

Source: Authors' own presentation.

It was observed that as the number of particles goes up, the number of required iterations goes down. In a similar fashion, if the particles are too spread at the beginning, it will take more iterations than if they were closer. Unlike deterministic algorithms (Floudas, 1999), this kind of approaches generates a different set of solutions for each run (in case the system has a solution). Should a given solution be required to improve, the search space can be tightened.

3.2. Ten equations system

As in the previous case, NSE (13) was implemented in C1. Table 4 summarizes the results for 100 particles.

$$
\begin{aligned}
F\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_a * \operatorname{sen}(x_b) \\
& + x_c^{x_d} - x_e * \tan(x_f) + 54 * x_g - 35 * x_n \\
& + x_i - 16 * x_j + 32 \\
G\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = 3 * x_a + 8 * x_b \\
& + x_c - x_d + x_e - x_f + 7 * x_g - 8 * x_n + x_i \\
& + x_j - 89 + x_c * \cos(x_b) - 54 \\
H\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_c * e^{x_a} + 3 * x_b \\
& + 6 * x_a + 91 + 3 * x_d + 10 * x_e + 3 * x_f - x_j \\
& + x_d^{\cos}(x_e) \\
I\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_a + 8 * x_b \\
& - 3 * x_c + x_d + 1 + x_g - x_i + 8 * x_j \\
& - 67 * \cos(x_b) + 5 * x_a^{x_d} \\
J\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_f + 3 * x_a \\
& - 6 * x_b + 8 * x_c - 3 * x_i \\
& - x_d * \operatorname{sen}(x_d) + x_g + 1 \\
K\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_g + x_a \\
& + 3 * x_b - 8 * x_i + x_c * \tan\left(\frac{xh}{xf}\right) + 54 \\
& + 3 * x_b - 8 * x_i + x_c * \tan\left(\frac{xh}{xf}\right) + 54 \\
L\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_b + 3 * x_j - 6 * x_e \\
& + x_d - x_a + x_j^{\frac{x_i}{x_b}} + x_d - x_a + x_e^{\frac{x_i}{x_b}} \\
M\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_i + 3 * x_b \\
& + 8 + x_c - x_j + x_i + \operatorname{sen}(x_a - x_e) - 98 \\
N\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = 4 * x_j - 8 * x_i \\
& + 32 * x_a - x_b + x_c * x_e^{\frac{x_i}{x_b}} + 32 * x_a - x_b + x_c * x_e^{\frac{x_i}{x_b}} \\
O\ & (x_a, x_b, x_c, x_d, x_e, x_f, x_g, x_h, x_i, x_j) = x_d + x_c - 10 * x_f - x_g - x_c
\end{aligned}
\tag{13}
$$

Tabla 4. Ten equations, 100 particles

Iter	Time	Result	Function evaluation
24448	1min 52s	Xa= -0.5277680826 - 1.5283098689i	F= -4.5448445007e-010 -2.8121434070e-009i
		Xb= 7.7962005980 - 1.4231054183i	G= 2.5054021080e-009 +1.1053923998e-008i
		Xc= 14.4156920826 - 6.3345255965i	H=-6.6833618639e-009 -6.0228955156e-010i
		Xd= -8.9800859158 - 2.3314054570i	I=-4.9275434164e-009 +1.2331246207e-008i
		Xe= -2.4629686688 + 3.2381025794i	J= -2.1878285850e-009 -2.5049473606e-010i
		Xf= -17.2730895444 - 0.0029571587i	K= -9.1274046099e-009 +1.1580755910e-008i
		Xg= -8.2930036068 + 2.3284482007i	L= -1.9023227438e-009 -5.3774200648e-009i
		Xh= -16.8858945561 + 5.2451585585i	M= -1.5188561520e-010 +2.8231426086e-009i
		Xi= 3.8674931214 - 2.7606840329i	N=-4.2459404881e-009 -4.1239260895e-009i
		Xj= 2.4340896095 + 4.8108047237i	O= 2.1884927648e-008 -9.7565134638e-008i
19758	1min 29s	Xa= -0.4899906343 - 1.4390721752i	F= 1.2253373427e-008 -1.1770282526e-009i
		Xb= 1.2105178720 - 2.4443423591i	G= -2.9848820304e-008 -3.3414693234e-010i
		Xc= -0.5709205322 -17.0521974483i	H= 1.46152956404e-009 +1.3261432841e-008i
		Xd= -3.8941329032 - 3.8089631076i	I= -1.4613814202e-008 -2.7360129613e-008i
		Xe= -2.4547751166 + 3.8140058133i	J= 9.7342578442e-011 -1.4803077874e-008i
		Xf= 7.6648459103 - 3.1160236655i	K= 2.8504723914e-008 +4.5085045918e-008i
		Xg= 11.5589788969 + 0.6929393892i	L= 5.0908277593e-008 -1.9243771021e-009i
		Xh= 2.4422089736 - 4.5840164960i	M= -5.2479123269e-009 +1.2282612261e-008i
		Xi= 1.4352243372 - 1.8927927261i	N= -3.9155239051e-009 -2.2080389073e-008i
		Xj= -3.8122143647 + 8.5176659594i	O= 8.3430992670e-008 -5.2982098708e-008i

Source: Authors' own presentation.

It is observed that by increasing the size of the NSE, as well as the number of particles, the required computer resources increase almost exponentially. For every case, the number of iterations was in the same range, even though they are inversely proportional to the number of particles. Therefore, the following experiments were performed on C2.

3.3. System of five equations in the cluster

NSE (12) was implemented in C2 and the results are summarized in tables 5 and 6.

Table 5. Five equations, 10 particles (using C2)

Iter.	Time	Result	Function evaluation
3400	3 s	Xa= 0.3838079634 + 6.1948786017i	F= -5.7142762716e-008 -5.9558418641e-008i
		Xb= 7.9225469908 - 1.0489956145i	G= -2.5578124507e-008 -9.6165717167e-008i
		Xc= 13.4409372199 -27.7778222274i	H= 3.3855009463e-008 +6.7293625783e-008i
		Xd= -31.797461044 + 2.5669947032i	I= 8.0641592959e-008 -4.0193143036e-008i
		Xe= 0.0025668988 + 0.3105349073i	J= 6.6515632113e-008 +1.6892528265e-008i
2053	2 s	Xa= 0.9383674408 + 2.1299016662i	F= 1.4610113119e-008 +9.5705505743e-008i
		Xb= 4.6842604783 - 1.4636484313i	G= 7.8593998865e-008 +2.5040030494e-008i
		Xc= 15.0954367479 +44.1193856691i	H= 5.2103377168e-008 +8.3392221129e-008i
		Xd= 1.5293410296 + 4.9835891637i	I= 6.3720266154e-008 -3.2842034019e-008i
		Xe= -0.0839990018 - 0.1311891326i	J= -6.3679550522e-009 +9.0587263912e-008i

Source: Authors' own presentation.

It can be seen that by increasing the amount of nodes, the run times are almost identical. Generally, it was observed that by using more particles, the average solution time also increased, while the number of iterations decreased. In most cases, this numerical approximation has a high convergence speed and provides satisfactory solutions.

Table 6. Five equations, 1000 particles (using C2)

Iter.	Time	Result	Function evaluation
647	39 s	Xa= 2.2449516499 + 3.7341905866i	F= -2.8481523806e-008 +5.7738451131e-008i
		Xb= 13.0650469983 - 0.4456976616i	G= -8.9964004246e-008 +2.7700457261e-008i
		Xc= 11.8494401963 - 4.1722507786i	H= 3.3100064911e-008 +3.7518764096e-008i
		Xd= -7.4348951904 + 2.0739670173i	I= 1.7868227434e-008 -5.7812462741e-008i
		Xe= 0.0841438795 + 0.0330677417i	J= -6.5719174103e-008 +6.0693988502e-008i
626	37 s	Xa= 2.2449516499 + 3.7341905857i	F= -6.2151654490e-008 +4.3906756453e-008i
		Xb= 13.0650469968 - 0.4456976615i	G= -6.8321966751e-008 +3.6126424963e-008i
		Xc= 11.8494402091 - 4.1722507714i	H= -5.4405157712e-008 -2.3531416815e-009i
		Xd= -7.4348951939 + 2.0739670128i	I= -8.8861543487e-008 -3.9075453669e-008i
		Xe= 0.0841438805 + 0.0330677419i	J= 6.0724950401e-008 -6.2071759288e-008i

Source: Authors' own presentation.

4. Observations and Conclusions

It is evident that it is of great help to transform a problem of solution of a NSE into an optimization one, especially for finding its real and complex roots. Even though this strategy is well known in literature, a mathematical demonstration

is provided here. The modified PSO used during this research, is an efficient optimization method that can be easily implemented to find the solution of these systems.

For the cases studied here, the fact of increasing the amount of particles, increased the run time, but a general correlation was not found. Moreover, and because it is a stochastic method, results are not expected to be fully repeatable. However, the solutions found by this algorithm could be feed to traditional deterministic ones.

A proper choice of the amount of particles, as well as their initial position, provided an increased convergence speed. Regarding the cluster, it is not enough to simply have the hardware, storage and networking resources, but the algorithm has to be parallelizable, which could not be done here. It was found too that the convergence speed depends on both, the size of the NSE and the complexity of its equations. A ten equation system was tried to solve in the cluster, but due to the high RAM requirements, the program did not run.

References

AIJIA, O.; YONGQUAN, Z. and QIFANG, L. *Hybrid particle swarm optimization algorithm for solving systems of nonlinear equations.* Nanning, China: Guangxi University for Nationalities, Granular Computing, 2009, GRC '09.

BIANCHINI, M.; FANELLI, S. and GORI, M. Optimal algorithms for well-conditioned nonlinear systems of equations. *IEEE Trans On Computers*. 2001, vol. 50, no. 7, pp. 689-698.

BRITS, R.; ENEGELBRECHT, A. and VAN DEN BERGH, F. *Solving systems of unconstrained equations using particle swarm optimization*. Pretoria, South Africa: Department of Computer Science, University of Pretoria, 2002.

CLERC, M. *Particle swarm optimization.* 1st Edition. ISTE, 2006. Chap. 3

CUI, Z. and CAI, X. *Using social cognitive optimization algorithm to solve nonlinear equations.* Shanxi, China, 2010. Proc. 9th IEEE Int. Conf. On Cognitive Informatics, June 2005. pp. 199-203.

FLOUDAS, C. *Deterministic global optimization*. Kluwer Academic Publisher, 1999. Chap. 12.

GENG H. T.; SUN Y. J.; SONG Q. X. *et al.* Research of ranking method in evolution strategy for solving nonlinear system of equations. The 1st International Conference on Information Science and Engineering (ICISE2009), IEEE Computer Society, pp. 348-351.

GÓMEZ, L. *Propuesta de demostración del teorema sobre la relación entre sistemas de ecuaciones y el problema de optimización* (Comunicación interna UIS. Nov 11, 2010).

GROSAN, C. and ABRAHAM, A. A new approach for solving nonlinear equations systems. *IEEE Trans. On Systems and Cybernetics-Part A: Systems and Humans*. 2008, vol. 38, no. 3, pp. 698-713.

GROSAN, C. and ABRAHAM, A. Multiple solutions for a system of nonlinear equations. *International J. of Innovative Computing. Information and Control, ICIC.* 2008, pp. 76-82.

HATANAKA, T; UOSAKI, K. and KOGA, M. *Evolutionary computation approach to block oriented nonlinear model.* Control Conference. 2004. 5th Asian, pp. 90-96.

HUI, W. and ZHAO, Z. *A neural network algorithm for solving systems of nonlinear equations*. Changsha, Hunan, China: College of Electrical & Information Engineering, University of Science & Technology, 2008.

LUO, Y. Z.; TANG, G. J. and ZHOU, L. N. Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method. *Applied Soft Computing*. 2008, 8, pp. 1068-1073.

ORTEGA, J. and RHEINBOLDT, W. *Iterative solution of nonlinear equations in several variables*. New York: Academic Press, 1970.

OUYANG, A.; ZHOU, Y. and LUO, Q. *Hybrid particle swarm optimization algorithm for solving systems of nonlinear equations.* Granular Computing, 2009, GRC '09. pp.46-465.

PARSOPOULOS, K. and VRHATIS, M. *Particle swarm optimization and intelligence: advances and applications.* Ed. Information Science Reference, 2010, Chaps 1-4.

RAO, S. *Engineering optimization, theory and practice.* 4th edition. Ed. John Wiley & Sons, 2009. Chap. 2-13.

TSOULOS, I. and STAVRAKOUDIS, A. On locating all roots of systems of nonlinear equations inside bounded domain using global optimization methods. *Nonlinear Analysis: Real World Applications*. 2010, vol. 11, no. 4, pp 2465-2471.

YANG, B.; ZHANG, Z. and SUN Z. Computing nonlinear lst estimator based on a random differential evolution strategy. *Tsinghua Science and Technology.* 2008, vol. 13, no.1, pp.1007-0214.